Санкт-Петербургский государственный университет
**Кафедра механики управляемого движения**

# Козлов Андрей Андреевич

**Выпускная квалификационная работа бакалавра**

# Движение космического манипуляционного робота в среде с препятствиями

100400
Прикладная математика и информатика

**Научный руководитель:**
кандидат физ.-мат. наук
доцент
Шиманчук Д. В.

**Санкт-Петербург**
**2017**

# Содержание

# 1 Введение

В связи с освоением и изучением Солнечной системы человечеством возникает необходимость в использовании разного рода автоматизированных и автоматических устройств. Примером уже используемых автоматизированных решений могут выступать марсоходы Curiosity и Opportunity используемые NASA. Однако представляют интерес не только роботы, способные изучать планеты, но и устройства для работы на космических станциях в условиях невесомости. Примером такого робота может выступать Personal Satellite Assistant также представленный NASA [1].

В связи с актуальностью обозначенных задач космической робототехники, в этой работе исследуется свободно-летающий космический робот манипулятор, рассматриваемый как объект маломерной космической техники. В механическом смысле данный робот представляет собой систему состоящую из твердого тела (основания) и шарнирно связанными с ним звеньями [4]. Средой применения такого робота является окрестность космической станции в которой он выполняет следующие функции:

1. Ремонт и обслуживание космической станции, с целью минимизировать риск для жизни человека связанный с необходимостью выполнять монтажные работы снаружи станции.

2. Идентификация параметров космических объектов с целью определения оптимальной стратегии управления данными объектами [4].

3. Применение космического манипуляционного робота, как экспериментальной платформы для проведения операций за пределами космической орбитальной станции, в связи с невозможностью выполнить некоторые операции на борту станции, таких как: влияние магнитных и электростатических полей, различных вибраций и других возмущающих факторов [4].

Для выполнения указанных функций в программном комплексе отвечающем за движение робота возникает необходимость использовать алгоритм движения в среде с препятствиями, с целью исключить столкновение

робота с компонентами космической станции и объектами внутри нее. Такие алгоритмы разрабатывались с 60-х годов прошлого века (A, A*, theta*) и активно используются по сей день в различных системах (марсоходы, автомобили с автопилотом, компьютерные игры).

В данной работе рассматривается применение RRT алгоритма (Rapidly exploring random trees) [5] для построения траектории обеспечивающей обход препятствий применительно к модели космического робота манипулятора [4], что в конечном итоге позволяет получить функции управления данным объектом.

С учетом специфики рабочей среды можно сделать следующие допущения относительно математической модели, которые упрощают исследование:

1. Достаточно высокая орбита космической станции и размеры робота позволяют пренебречь силой тяжести.

2. Безвоздушное пространство позволяет пренебречь силами сопротивления среды.

# 2 Постановка задачи

Рассматриваемая модель представляет собой несущее тело с присоединенным к нему манипулятором, который представляет систему шарнирно связанных между собой звеньев длиной $r_1$ (плечо) и $r_2$ (локоть), предполагается, что массы звеньев пренебрежительно малы (см. рис. 1).

Для моделирования управляемого движения данной механической системы в среде с препятствиями необходимо решить следующие подзадачи:

1. Вывод уравнений движения рассматриваемой механической системы
2. Построение возможных точек траекторий движения центра масс основания робота
3. Планирование траектории
4. Оценка необходимых управляющих воздействий

Уравнения движения системы будем искать в виде уравнений Лагранжа второго рода

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_j} - \frac{\partial T}{\partial q_j} = F_j(t), \quad j = \overline{1, n}.$$

Для получения возможных точек траектории движения основания робота используем алгоритм быстро-исследующий случайных деревьев RRT, результатом работы которого будет кусочно-линейная траектория, обеспечивающая обход препятствий из начального положения в конечное. Далее на основании результатов работы алгоритма представляется возможным дальнейшее планирование траектории движения основания с необходимой степенью гладкости. Для оценки управляющих воздействий решается обратная задача динамики.

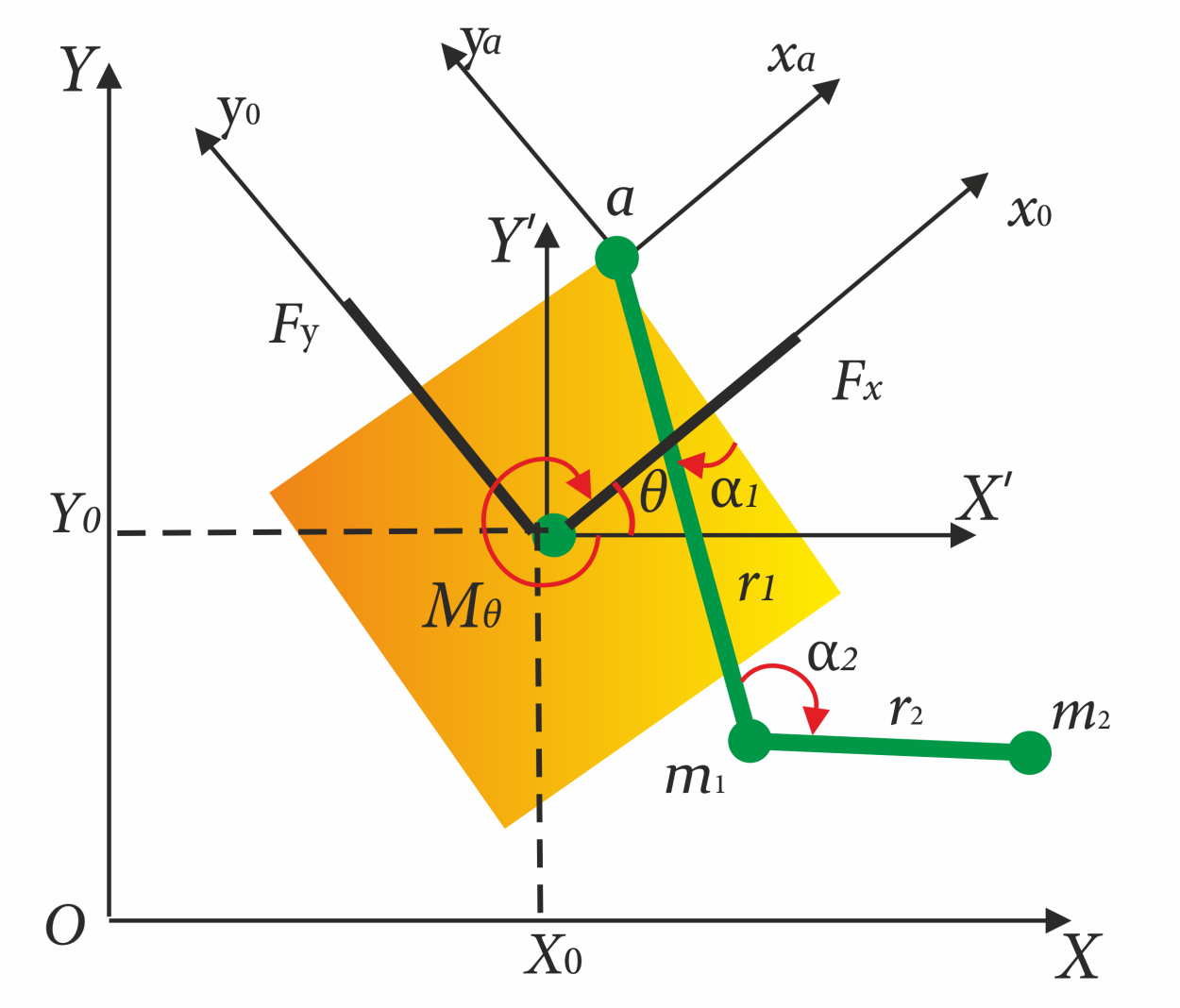


Рис. 1: Механическая модель

# 3 Обзор литературы

Подход к управлению космическим манипуляционным роботом рассматривается в статьях [2] - [3]. В них раскрывается проблема синтеза оптимальной механической структуры данного робота по критериям:

1. Достижимости статической сбалансированности конфигурации космического модуля как с захваченным полезным грузом, так и без него

2. Возможности обеспечения минимального значения момента инерции в фазе полета

В статье [4] решена задача вывода уравнений движения космического манипуляционного робота. В данной работе будет рассмотрена упрощенная модель [4], в частности, подобная задача будет решена в плоскости с количеством степеней свободы положения равным пяти.

В статье [5] рассмотрен алгоритм обхода препятствий RRT, который будет реализован в данной работе для планирования траектории. Данный алгоритм может быть улучшен применением эвристических функций рассмотренных в [6].

# 4 Построение математической модели

Рассматриваемая модель представляет собой несущее тело с присоединенным к нему манипулятором, который представляет систему шарнирно связанных между собой звеньев длиной $r_1$ (плечо) и $r_2$ (локоть), предполагается, что массы звеньев пренебрежительно малы (см. рис. 1). Введем системы координат:

1. $OXY$ - базовая (иннерциальная) система координат. Такая система координат связана с космической станцией.

2. $oX'Y'$ - подвижная система координат с центром в центре масс несущего тела и осями, параллельными осям абсолютной системы координат.

3. $ox_0y_0$ — система координат, жестко связанная с несущим телом.

4. $ax_ay_a$ – координатная система, определяющая место присоединения манипулятора к корпусу космического манипуляционного робота, координаты которого находятся в точке $a(x_a, y_a)$.

Изменяемыми координатами являются:

1. положение полюса $o(X, Y)$, определяемое координатами точки ($X_o$,$Y_o$)

2. угловое отклонение $\theta$ в осях базовой системы координат

3. углы $\alpha_1$, $\alpha_2$, которые определяют положение звеньев манипулятора

Данная модель представляет собой свободную механическую систему, подчиняющуюся внутренним голономным неосвобождающим стационарным связям. Положим, что отсутствуют силы сопротивления среды, а невесомость позволяет не учитывать силы тяжести [4]. У данной системы пять степеней свободы положения. Для получения уравнений движения используем уравнения Лагранжа второго рода в виде

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_j} - \frac{\partial T}{\partial q_j} = F_j(t), \quad j = \overline{1,n}. \tag{1}$$

## 4.1 Вывод функции кинетической энергии

Из вида уравнений Лагранжа второго рода и сделанных выше допущений понятно, что для определения уравнений движения необходимо построить функцию кинетической энергии абсолютного движения как функцию обобщенных коодинат $q_1 = X_0, q_2 = Y_0, q_3 = \theta, q_4 = \alpha_1, q_5 = \alpha_2$.

Для любого момента времени $t$ положение механической системы на плоскости $OXY$ для несущего тела задается координатами $X_0, Y_0, \theta$, а положение звеньев манипулятора представляется точками $m_1, m_2$. Их проекции на оси связной системы координат $ox_0y_0$ определяются координатами $x_i^0, y_i^0$. Учитывая, что при введении координат положение полюса $o$ было совмещено с центром масс несущего тела, можно выписать проекции центра масс всей системы на те же оси:

$$x_c^0 = \frac{\sum_{i=0}^{2} m_i x_i^0}{M},$$

$$y_c^0 = \frac{\sum_{i=0}^{2} m_i y_i^0}{M}.$$

Дифференцируя по времени, найдем проекции скорости центра масс на оси связной системы координат:

$$\dot{x_c^0} = \frac{\sum_{i=0}^{2} m_i \dot{x_i^0}}{M},$$

$$\dot{y_c^0} = \frac{\sum_{i=0}^{2} m_i \dot{y_i^0}}{M}.$$

Где $M = \sum_{i=0}^{2} m_i$ - полная масса системы.

Теперь выпишем координаты, определяющие положение точек механической системы проекциями на оси подвижной системы координат:

$$x_i = x_i^0 cos\theta - y_i^0 sin\theta,$$

$$y_i = y_i^0 cos\theta + x_i^0 sin\theta.$$

Тогда, путем дифференцирования можно определить проекции скоростей в той же системе отсчета:

$$\dot{x_i} = \dot{x_i^0}cos\theta - \dot{y_i^0}sin\theta - \dot{\theta}(y_i^0 cos\theta + x_i^0 sin\theta),$$

$$\dot{y_i} = \dot{y_i^0}cos\theta + \dot{x_i^0}sin\theta - \dot{\theta}(x_i^0 cos\theta - y_i^0 sin\theta).$$

Проделав аналогичные действия для координат центра масс, получим их проекции и проекции их скоростей на оси подвижной системы координат:

$$x_c = x_c^0 cos\theta - y_c^0 sin\theta,$$

$$y_c = y_c^0 cos\theta + x_c^0 sin\theta,$$

$$\dot{x_c} = \dot{x_c^0}cos\theta - \dot{y_c^0}sin\theta - \dot{\theta}(y_c^0 cos\theta + x_c^0 sin\theta),$$

$$\dot{y_c} = \dot{y_c^0}cos\theta + \dot{x_c^0}sin\theta - \dot{\theta}(x_c^0 cos\theta - y_c^0 sin\theta).$$

Теперь, согласно теореме Кёнига, можно выразить функцию кинетической энергии абсолютного движения системы как сумму функций кинетических энергий движения центра масс и движения относительно центра масс:

$$T = \tfrac{1}{2}m_\Sigma(\dot{X_0}^2 + \dot{Y_0}^2) + \tfrac{1}{2}I_0\theta^2 + \tfrac{1}{2}\sum_{i=1}^{2} m_i[(\dot{x_i^0} - y_i^0\dot{\theta})^2 +$$

$$+(\dot{y_i^0} + x_i^0\dot{\theta})^2] + (\dot{X_0}cos\theta + \dot{Y_0}sin\theta)\sum_{i=1}^{2} m_i(\dot{x_i^0} - y_i^0\dot{\theta}) +$$

$$+(\dot{Y_0}cos\theta + \dot{X_0}sin\theta)\sum_{i=1}^{2} m_i(\dot{y_i^0} - x_i^0\dot{\theta}). \qquad (2)$$

## 4.2 Вывод уравнений движения

После того как функция кинетической энергии абсолютного движения системы была найдена как фукция декартовых и угловых координат в базовой СК, необходимо представить ее как функцию от обобщенных координат. Для этого выпишем уравнения связывающие декартовы координаты с обобщенными:

$$x_1^0 = x_a - r_1 sin\alpha_1,$$

$$y_1^0 = x_a - r_1 cos\alpha_1,$$

$$x_2^0 = x_1^0 + r_2 sin(\alpha_1 + \alpha_2),$$

$$y_2^0 = y_1^0 + r_2 cos(\alpha_1 + \alpha_2).$$

После дифференцирования по времени данных уравнений получаем скорости точек системы

$$\dot{x_1^0} = -r_1\dot{\alpha_1} cos\alpha_1,$$

$$\dot{y_1^0} = r_1\dot{\alpha_1} sin\alpha_1,$$

$$\dot{x_2^0} = x_1^0 + r_2(\dot{\alpha_1} + \dot{\alpha_2})cos(\alpha_1 + \alpha_2),$$

$$\dot{y_2^0} = y_1^0 - r_2(\dot{\alpha_1} + \dot{\alpha_2})sin(\alpha_1 + \alpha_2).$$

Ускорения получаются путем повторного дифференцирования

$$\ddot{x_1^0} = -r_1(\ddot{\alpha_1} cos\alpha_1 - \dot{\alpha_1}^2 sin\alpha_1),$$

$$\ddot{y_1^0} = r_1(\ddot{\alpha_1} sin\alpha_1 - \dot{\alpha_1}^2 cos\alpha_1),$$

$$\ddot{x_2^0} = \ddot{x_1^0} + r_2((\ddot{\alpha_1} + \ddot{\alpha_2})cos(\alpha_1 + \alpha_2) - (\dot{\alpha_1} + \dot{\alpha_2})^2 sin(\alpha_1 + \alpha_2)),$$

$$\ddot{y_2^0} = \ddot{y_1^0} - r_2((\ddot{\alpha_1} + \ddot{\alpha_2})sin(\alpha_1 + \alpha_2) - (\dot{\alpha_1} + \dot{\alpha_2})^2 cos(\alpha_1 + \alpha_2)).$$

После того как выписаны уравнения связи декартовых координат с обобщенными можно переходить к выводу уравнений движения механической системы. Для этого подставим кинетическую энергию в виде (2) в уравнения Лагранжа второго рода (1). Для начала будем дифференцировать по обобщенным координатам $X_0$, $Y_0$ чтобы получить уравнения движения отвечающие линейным перемещениям основания. В итоге можно

записать полученные уравнения

$$M\ddot{X}_0 + \sum_{i=1}^{2} m_i[(\ddot{x_i^0}cos\theta - \ddot{y_i^0}sin\theta) - \ddot{\theta}(x_i^0 sin\theta + y_i^0 cos\theta) -$$
$$-2\dot{\theta}(\dot{x_i^0}sin\theta + \dot{y_i^0}cos\theta) + \ddot{\theta}^2(y_i^0 sin\theta - x_i^0 cos\theta)] = F_x cos\theta - F_y sin\theta, \qquad (3)$$

$$M\ddot{Y}_0 + \sum_{i=1}^{2} m_i[(\ddot{x_i^0}sin\theta + \ddot{y_i^0}cos\theta) + \ddot{\theta}(x_i^0 cos\theta + y_i^0 sin\theta) +$$
$$+2\dot{\theta}(\dot{x_i^0}cos\theta - \dot{y_i^0}sin\theta) - \ddot{\theta}^2(y_i^0 cos\theta + x_i^0 sin\theta)] = F_x cos\theta + F_y sin\theta. \qquad (4)$$

Теперь проделаем те же действия для обобщенной координаты $\theta$ чтобы получить уравнение отвечающее повороту основания робота. В итоге приходим к третьему уравнению движения

$$\ddot{\theta}[I_0 + \sum_{i=1}^{2} m_i({x_i^0}^2 + {y_i^0}^2)] - \ddot{X}_0 \sum_{i=1}^{2} m_i(x_i^0 sin\theta + y_i^0 cos\theta) + \ddot{Y}_0 \sum_{i=1}^{2} m_i(x_i^0 cos\theta -$$
$$-y_i^0 sin\theta) + \sum_{i=1}^{2} m_i[(x_i^0 \ddot{y_i^0} - y_i^0 \ddot{x_i^0}) + 2\dot{\theta}(x_i^0 \dot{x_i^0} - y_i^0 \dot{y_i^0})] = M_\theta. \qquad (5)$$

Теперь перейдем к последним обобщенным координатам $\alpha_1$, $\alpha_2$ и выпишем уравнения движения отвечающие перемещению манипулятора робота

$$\ddot{X}_0 \sum_{i=1}^{2} m_i \left(\frac{\partial x_i^0}{\partial q_j} cos\theta - \frac{\partial y_i^0}{\partial q_j} sin\theta\right) + \ddot{Y}_0 \sum_{i=1}^{2} m_i \left(\frac{\partial x_i^0}{\partial q_j} sin\theta + \frac{\partial y_i^0}{\partial q_j} cos\theta\right) +$$
$$+\ddot{\theta}_0 \sum_{i=1}^{2} m_i \left(x_i^0 \frac{\partial x_i^0}{\partial q_j} - y_i^0 \frac{\partial y_i^0}{\partial q_j}\right) + \sum_{i=1}^{2} m_i \left(\ddot{x_i^0} \frac{\partial x_i^0}{\partial q_j} + \ddot{y_i^0} \frac{\partial y_i^0}{\partial q_j}\right) =$$
$$= F_j^r(t) + Q_{jr}^{Cor} + Q_{jr}^{cf}, \qquad (6)$$

где обобщенные кориолисовы (cor) и центростремительные (cf) силы даны формулами

$$Q_{jr}^{cf} = -\dot{\theta}^2 \sum_{i=1}^{2} m_i(x_i^0 \frac{\partial x_i^0}{\partial q_j} + y_i^0 \frac{\partial y_i^0}{\partial q_j}),$$

$$Q_{jr}^{cor} = -2\dot{\theta} \sum_{i=1}^{2} m_i(x_i^0 \frac{\partial y_i^0}{\partial q_j} - y_i^0 \frac{\partial x_i^0}{\partial q_j}).$$

# 5 Моделирование управляемого движения

Наличие уравнений движения (3-6) позволяет перейти к задаче планирования траектории движения космического манипулционного робота в среде с препятствиями. Будем считать среду детерменированной, а препятствия в данной постановке задачи недвижимыми. Необходимо построить траекторию движения основания робота из начального положения в конечное обеспечивающую обход препятствий. Для построения такой тректории используем алгоритм быстро-исследующих случайных деревьев – RRT. В приложении к данной работе приведен листинг программы реализующей алгоритм на языке C#.

## 5.1 Описание реализации алгоритма

Идея данного алгоритма заключается в применении генератора случайных чисел (ГСЧ) для получения траектории движения. Из начального положения строится быстро исследующее случайное дерево, которое представляет собой граф, вершины которого хранят абсциссу и ординату точки на плоскости. Ребра данного графа суть отрезки фиксированной длины $\varepsilon$, соединяющие вершину «родителя» с вершиной «потомком». Данный граф строится из начального положения путем добавления вершин в дерево по следующему алгоритму:

```
G.init(q_start)
{
while q_current.distance(q_finish)>=accuracy
    q_rand = RAND_CONF()
    q_near = NEAREST_VERTEX(q_rand, G)
    q_new = NEW_CONF(q_near, q_rand, eps)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
    return G
}
```

где

1. `q_current` — текущая точка в графе
2. `q_rand` — точка выбранная ГСЧ
3. `NEAREST_VERTEX(q_rand, G)` — функция поиска ближайшей в дереве точки к заданной ГСЧ
4. `NEW_CONF(q_near, q_rand, eps)` — функция получения точки выбранной от ближайшей в графе в направлении точки, заданной ГСЧ на заданном расстоянии $\varepsilon$
5. `G.add_vertex(q_new)` — функция добавления вершины в граф
6. `G.add_edge(q_near, q_new)` — функция добавления ребра в граф

Препятствия задаются в виде замкнутой ломаной и реализованы в программе так же в виде графа. Обход препятствий реализуется с помощью добавления проверки на возможность пройти по данному маршруту в `NEW_CONF(q_near, q_rand, eps)`. Суть такой проверки заключается в следующем: пересекается ли ребро от вершины «родителя» до новой с гранью препятствия.

Результатом работы алгоритма является кусочно-линейная траектория, записанная в виде упорядоченного набора точек, позволяющего попасть из начального положения в конечное с учетом обхода препятствия (см. рис. 2 - 3). Такую траеторию лекго аппроксимировать до гладкой как две функции координат от времени $X_0(t), Y_0(t)$ для получения необходимой гладкости.

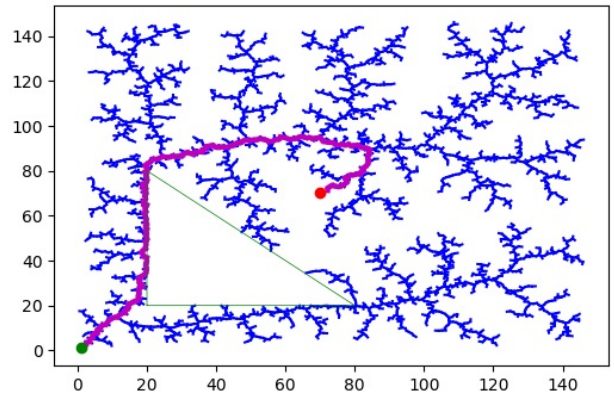


Рис. 2: Итераций всего - 3718. Длина пути - 199 точек. Время - 2.2 сек.

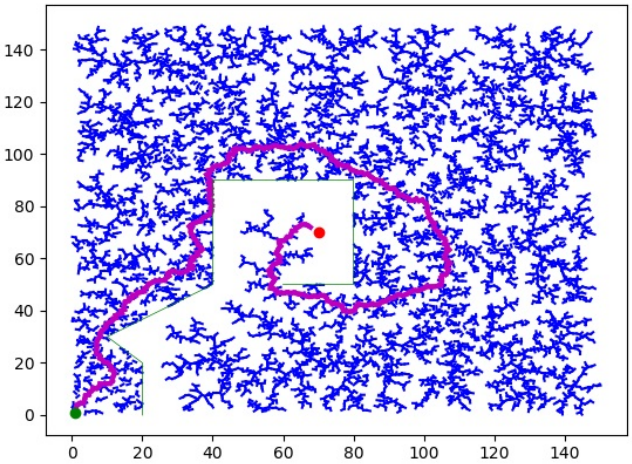


Рис. 3: Итераций всего - 9449. Длина пути - 343 точек. Время - 9.6 сек.

## 5.2 Пример решения обратной задачи динамики

Для примера решения обратной задачи динамики выберем закон движения представленный фиолетовым цветом на рис. 2. Данная траектория движения состоит из 199 точек и обеспечивает обход препятствия заданного зеленым треуголником с вершинами (20,20), (20,80), (80,20). Как отмечено выше, необходимо данную траекторию аппроксимировать гладкой функцией, например, полиномом пятого порядка. В пакете Matlab были получены такие аппроксимирующие полиномы в виде

$$X_0(t) = p_1^x t^5 + p_2^x t^4 + p_3^x t^3 + p_4^x t^2 + p_5^x t + p_6^x, \tag{7}$$

$$Y_0(t) = p_1^y t^5 + p_2^y t^4 + p_3^y t^3 + p_4^y t^2 + p_5^y t + p_6^y. \tag{8}$$

При

$$p_1^x = -2.532e^{-09}, I_1^x = (-3.051^{e-09}, -2.013e^{-09}).$$

$$p_2^x = 4.177e^{-07}, I_2^x = (1.582e^{-07}, 6.771e^{-07}).$$

$$p_3^x = 9.872e^{-05}, I_3^x = (5.184e^{-05}, 0.0001456).$$

$$p_4^x = -0.02033, I_4^x = (-0.02402, -0.01664).$$

$$p_5^x = 1.162, I_5^x = (1.043, 1.281).$$

$$p_6^x = -2.669, I_6^x = (-3.854, -1.484).$$

$$p_1^y = -1.623e^{-10}, I_1^y = (-6.01e^{-10}, 2.763e^{-10}).$$

$$p_2^y = 2.506e^{-07}, I_2^y = (3.129e^{-08}, 4.699e^{-07}).$$

$$p_3^y = -0.0001084, I_3^y = (-0.0001481, -6.882e^{-05}).$$

$$p_4^y = 0.01272, I_4^y = (0.009599, 0.01584).$$

$$p_5^y = 0.3735, I_5^y = (0.2727, 0.4743).$$

$$p_6^y = 3.275, I_6^y = (2.273, 4.276).$$

Где $I_i^x, I_i^y$ - доверительные интервалы.

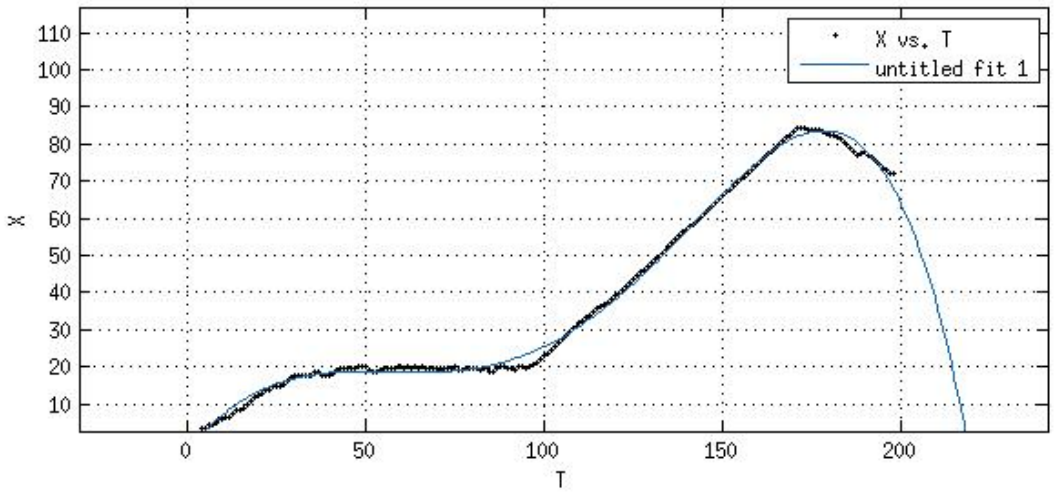


Рис. 4: Аппроксимация X(t)

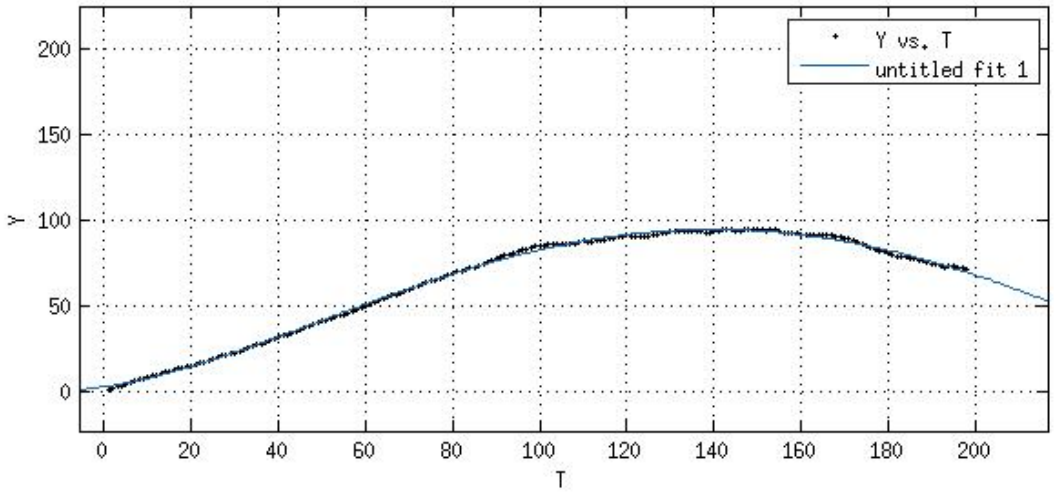


Рис. 5: Аппроксимация Y(t)

При наличии достаточно гладких функций, соответствующих обобщенным координатам, движения основания робота становится возможным определение управляющих воздействий путем подстановки найденных функций (7), (8) в систему (3)-(6). В виду того, что рассматривается задача о перемещении несущего тела, при фиксированом относительном положениеии манипулятора, т.е. $\alpha_1, \alpha_2$ являются константами. В таком случае уравнения движения принимают вид

$$M\ddot{X}_0 + \sum_{i=1}^{2} m_i[-\ddot{\theta}(x_i^0 sin\theta + y_i^0 cos\theta) +$$
$$+\ddot{\theta}^2(y_i^0 sin\theta - x_i^0 cos\theta)] = F_x cos\theta - F_y sin\theta, \qquad (9)$$

$$M\ddot{Y}_0 + \sum_{i=1}^{2} m_i[\ddot{\theta}(x_i^0 cos\theta + y_i^0 sin\theta) +$$
$$+\ddot{\theta}^2(x_i^0 sin\theta + y_i^0 cos\theta)] = F_x cos\theta + F_y sin\theta, \qquad (10)$$

$$-\ddot{X}_0 \sum_{i=1}^{2} m_i(x_i^0 sin\theta + y_i^0 cos\theta) + \ddot{Y}_0 \sum_{i=1}^{2} m_i(x_i^0 cos\theta - y_i^0 sin\theta) +$$
$$+\ddot{\theta}[I_0 + \sum_{i=1}^{2} m_i(x_i^{0^2} + y_i^{0^2})] = M_\theta. \qquad (11)$$

Предположим, что манипулятор прикреплен к основанию таким образом, что его центр масс лежит в начале относительной системы координат $ox_0y_0$, т.е совпадает с центром масс основания. Тогда будет иметь место поступательное движение робота ($\theta = const$), что следует из неуправляемого уравнения (11), когда $M_\theta = 0$. С учетом подстановки (7), (8), уравнения (9), (10) принимают вид

$$M(20p_1^x t^3 + 12p_2^x t^2 + 6p_3^x t + 2p_4^x) = F_x cos\theta - F_y sin\theta,$$

$$M(20p_1^y t^3 + 12p_2^y t^2 + 6p_3^y t + 2p_4^y) = F_x sin\theta + F_y cos\theta.$$

Из данной системы можно выписать управляющие воздейсвия в явном виде

$$F_x = Mcos\theta(20p_1^x t^3 + 12p_2^x t^2 + 6p_3^x t + 2p_4^x) +$$
$$+Msin\theta(20p_1^y t^3 + 12p_2^y t^2 + 6p_3^y t + 2p_4^y), \qquad (12)$$

$$F_y = -Msin\theta(20p_1^x t^3 + 12p_2^x t^2 + 6p_3^x t + 2p_4^x) +$$
$$+Mcos\theta(20p_1^y t^3 + 12p_2^y t^2 + 6p_3^y t + 2p_4^y). \qquad (13)$$

На рис. 6, 7 представлены графики управляющих воздействий (12), (13) при $\theta = 0$.

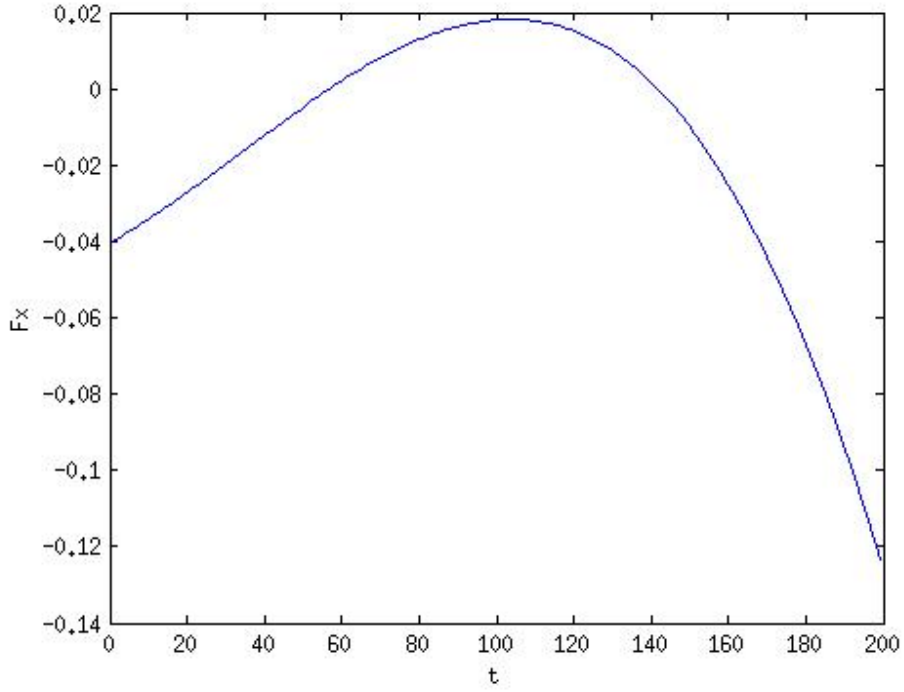


Рис. 6: График управляющего воздействия Fx(t) для конфигурации M=1, $\theta = 0$

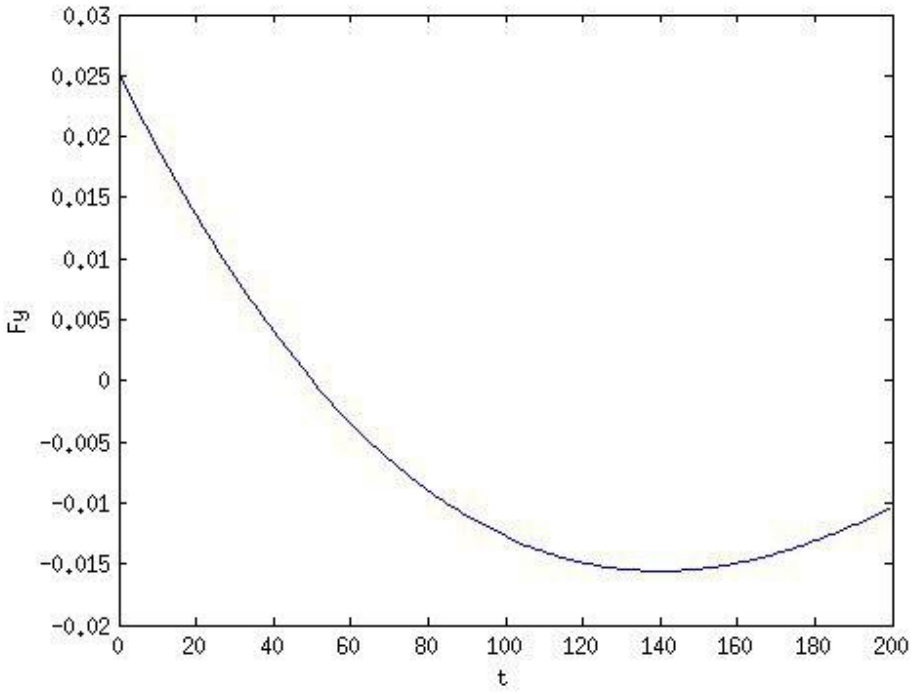


Рис. 7: График управляющего воздействия Fx(t) для конфигурации M=1, $\theta = 0$

# 6 Выводы

Относительное движение тел составляющих рассматриваемую механическую систему выражается наличием обобщенных кориолисовых и центробежных сил в представленных уравнениях движения. Отметим также, что принятые допущения исключают появление обобщенных сил, соответствующих силам тяжести и диссипативным силам. Более того, постановка задачи позволяет свести систему пяти уравнений второго порядка к системе трех уравнений второго порядка, поскольку в рассматриваемой задаче относительное движение отсутствует – определяется траектория основания робота в среде с препятствиями, когда манипулятор неподвижен.

При решении задачи поиска возможных точек траектории движения робота было замечено, что при дискретизации пространства сходимость алгоритма RRT гарантирована возможностью выполнить полный перебор, если решение существует, т. е. можно построить путь из начального фазового состояния космического манипуляционного робота в его конечное положение. В противном случае, выполнение полного перебора автоматически показывает отсутствие решения для заданной сетки.

# 7 Заключение

По итогам проделанной работы:

1. Представлен вывод уравнений управляемого движения космического манипуляционного робота как свободной механической системы состоящей из основания и шарнирно прикрепленного двухзвенного манипулятора

2. Решена задача поиска возможных точек траектории движения робота в среде с препятствиями

3. На основании результата работы алгоритма RRT решена задача планирования траектории движения основания робота

4. Для оценки управляющих воздействий решена обратная задача динамики роботов

5. В приложении представлен листинг программы реализующей алгоритм RRT. Результатом выполнения программы является кусочно-линейная траектория движения робота

# 8 Приложение

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
using System.IO;


namespace ConsoleApplication1
{
    class vector
    {
        public double cx, cy;

        public vector()
        {

        }
        public static vector vectinit(point a, point b)
        {
            vector vct = new vector();
            vct.cx = b.x - a.x;
            vct.cy = b.y - a.y;
            return vct;
        }


    }
    class point
    {
```

```
public double x, y;
public double prex, prey;
public point parent;
public bool islast;
public point(double coordX, double coordY,
 double preX, double preY, bool lastflag)
{
    x = coordX;
    y = coordY;
    prex = preX;
    prey = preY;
    islast = lastflag;
}

public void writefp(StreamWriter fs)
{
    string str = String.Concat(this.x.ToString()
    , " ", y.ToString(), Environment.NewLine);
    str.Replace(",", ".");
    fs.Write(str);
}
public void setParent(point par)
{
    parent = par;
}
public void show()
{
    Console.WriteLine("point X=" + this.x +
     " Y=" + this.y);
}
public point()
{
```

```
    }

    public static double distance(point point1, point point2)
    {
        return Math.Sqrt((point2.x - point1.x) *
        (point2.x - point1.x) +
         (point2.y - point1.y) * (point2.y - point1.y));
    }
}

class obstacle
{
    public int anglesnum;
    public List<point> points = new List<point>();

    public obstacle()
    {

    }
    public void show()
    {
        int i = 0;
        while (i != this.anglesnum)
        {
            this.points.ElementAt(i).show();
            i++;
        }
    }

    public void init()
    {
        Console.WriteLine("Enter number of angles:");
        this.anglesnum = Convert.ToInt16(Console.ReadLine());
```

```
            int i = 0;
            while (i != anglesnum)
            {
                point currentp = new point();
                Console.WriteLine("X"+i+":");
                currentp.x = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("Y"+i+":");
                currentp.y = Convert.ToDouble(Console.ReadLine());
                //currentp.setParent(currentp);
                this.points.Add(currentp);
                i++;
            }
        }
        public static void mapinit()
        {

        }
    }
        class RRT
        {
            public const double lambda = 1;

        public static int veccompos(vector v1, vector v2)
        {
            if ((v1.cx * v2.cy - v2.cx * v1.cx) > 0)
                return -1;
            else
                return 1;
        }



        public static bool collision_p(point a1, point a2,
         point b1, point b2)
```

```
{
    double v1 = (b2.x - b1.x) * (a1.y - b1.y)
    - (b2.y - b1.y) * (a1.x - b1.x);
    double v2 = (b2.x - b1.x) * (a2.y - b1.y)
    - (b2.y - b1.y) * (a2.x - b1.x);
    double v3 = (a2.x - a1.x) * (b1.y - a1.y)
    - (a2.y - a1.y) * (b1.x - a1.x);
    double v4 = (a2.x - a1.x) * (b2.y - a1.y)
    - (a2.y - a1.y) * (b2.x - a1.x);
    if ((v1 * v2 < 0) && (v3 * v4 < 0))
        return true;
    else
        return false;
}
public static bool collision_o(obstacle obs, point cp)
{
    int i = 0;
    point cur1 = new point();
    point cur2 = new point();
    bool flag = false;
    while ((i != obs.anglesnum-1)&&(flag==false))
    {
        cur1 = obs.points.ElementAt(i);
        cur2 = obs.points.ElementAt(i+1);
        flag = RRT.collision_p(cur1,cur2,cp.parent,cp);
        i++;
    }
    return flag;
}
public static point newpoint(point p1, point p2)
    {
        point newp = new point();
        newp.x = Math.Round(p1.x + lambda * (p2.x - p1.x) /
```

```
            Math.Sqrt((p2.x - p1.x) *
              (p2.x - p1.x) + (p2.y - p1.y) *
              (p2.y - p1.y)), 2);
            newp.y = Math.Round(p1.y + lambda * (p2.y - p1.y) /
             Math.Sqrt((p2.x - p1.x) *
              (p2.x - p1.x) + (p2.y - p1.y) *
              (p2.y - p1.y)), 2);
            return newp;
        }
        public static point rndmpoint(Random rand)
        {
            point randpt = new point();

            double min = 0, max = 150;
            randpt.x = Math.Round(min + rand.NextDouble() *
             (max - min), 2);
            randpt.y = Math.Round(min + rand.NextDouble() *
             (max - min), 2);
            return randpt;
        }
    }

    class tree
    {
        public List<List<point>> treelist;
        public int Step;
        public tree()
        {
            treelist = new List<List<point>>();
            Step = 0;
        }
        public List<point> this[int index]
        {
```

```
        get { return this.treelist[index]; }
        set { this.treelist[index] = value; }
    }
    public void addStep(point p)
    {
        List<point> thislist = new List<point>();
        this.treelist.Add(thislist);
        thislist.Add(p);
    }
    public void addPoint(point p, int Step)
    {
        treelist[Step].Add(p);
    }
    public point findCloser(point it)
    {
        int i = treelist.Count() - 1;
        int j = 0;
        point result = new point();
        double minDistance = 100000000.0;
        while (i >= 0)
        {
            j = 0;
            while (j < treelist[i].Count())
            {
                if (minDistance >
                point.distance(it, treelist[i][j]))
                {
                    result = treelist[i][j];
                    minDistance =
                    point.distance(it, treelist[i][j]);
                }
                j++;
            }
```

```
            i--;
        }
        return result;
    }
    public void treeInit(point start, point finish,
     double accuracy, obstacle obs)
    {
        tree RRTtree = new tree();
        point currentp = start;
        RRTtree.addStep(start);
        bool flag=false;
        int i = 1;
        var rand = new Random();
    while (point.distance(finish, currentp) > accuracy)
        {

            point randompoint = RRT.rndmpoint(rand);
            point closest = RRTtree.findCloser(randompoint);
            randompoint = RRT.newpoint(closest, randompoint);

        if (currentp.parent != closest)
            {
            randompoint.parent = closest;
            flag = RRT.collision_o(obs, randompoint);
            if (flag ==false)
                {
                closest.islast = false;
                randompoint.islast = true;
                RRTtree.addPoint(randompoint, 0);
                currentp = randompoint;
                i++;
                Console.WriteLine("Point: " + currentp.x +
                " " + currentp.y + "
```

```
                    || Parent: " + currentp.parent.x + " "
                    + currentp.parent.y);
                    }
                }
            }
            Console.WriteLine("Number of iterations: " + i);
            Console.WriteLine("Here is your path:");
            List<point> path = new List<point>();
            point backcurr = currentp;
            path.Add(backcurr);
            int j = 1;
            using (System.IO.StreamWriter file =
            new System.IO.StreamWriter(@"F:\Test.txt"))
            while (backcurr != start)
            {
                Console.WriteLine(backcurr.x +
                " " + backcurr.y);
                backcurr.writefp(file);
                path.Add(backcurr.parent);
                backcurr = backcurr.parent;
                j++;
            }

            Console.WriteLine("Number of points in the way: " + j)
        }
    }
}
```

# Список литературы


[1] https://www.nasa.gov/

[2] Богомолов В. П., Рутковский В. Ю., Суханов В. М. Проектирование оптимальной механической структуры свободнолетающего космического робототехнического модуля как объекта автоматического управления 1// Автомат. и телемех., 1998. № 5, С. 27—40.

[3] Богомолов В. П., Рутковский В. Ю., Суханов В. М. Проектирование оптимальной механической структуры свободнолетающего космического робототехнического модуля как объекта автоматического управления 2// Автомат. и телемех., 1998. № 6, С. 75—88.

[4] Глумов В. М., Земляков С. Д., Рутковский В. Ю., Суханов В. М. Оперативный компьютерный вывод и декомпозиция уравнений движения космического модуля // Автомат. и телемех., 2006. Вып. 1. С. 89–116.

[5] LaValle S. M., Kuffner J. J. Randomized kinodynamic planning // The International Journal of Robotics Research. 2001. P. 378–400.

[6] Naderi K., Rajamaki J., Hamalainen P. RT-RRT*: a real-time path planning algorithm based on RRT*// 8th ACM SIGGRAPH Conference on Motion in Games (MIG '15). ACM, New York, NY, USA, P. 113–118.